Project 5: IMDB Movie Analysis

Project Description:

The project aims at deriving insights from an IMDB movies dataset by answering various questions and performing analysis on the dataset. The project aims at teaching how to clean a dataset and handle the data using excel/python/R and then how to use these technologies for performing analysis and form a problem statement using 5 whys approach and then finding a solution.

Approach:

The approach to do this project is pretty straight forward. I started out with the data cleaning process since the provided dataset is huge and such huge datasets tend to have problems like missing data, duplicate data and wrong format. After all the cleaning process, I took a look at the questions asked and found a way to answer them after performing analysis on the new clean data. For all these tasks I used Jupyter Notebook i.e. python programming language.

Tech stack used:

- Jupyter Notebook
- Excel
- Google drive

Insights:

A. **Cleaning the data::** PThis is one of the most important step to perform before moving forward with the analysis. Use your knowledge learned till now to do this. (Dropping columns, removing null values, etc.)
Your task: Clean the data

Steps for doing this:

1) Imported the libraries NumPy and pandas and imported the dataset. Took a look into the information about the dataset.
2) Finding percentages of null values present in the columns of the dataset. Will use this in upcoming steps.
3) Dropping unnecessary columns. Many of the columns were not needed so removed those.
4) Drop unnecessary rows using columns with high null value percentage. The gross and budget had great percentages of null values in them.

5) Removed duplicate rows.

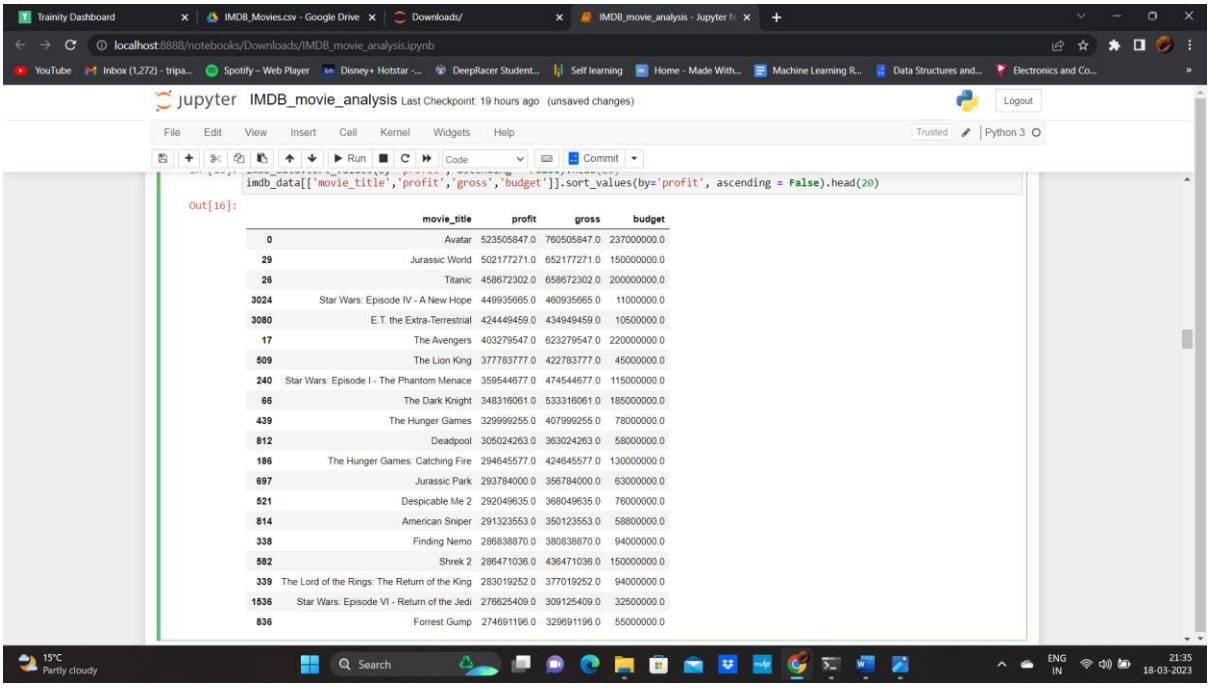
Here are the results: size of dataset

Before cleaning: 5043 x 28

After cleaning: 3849 x 13

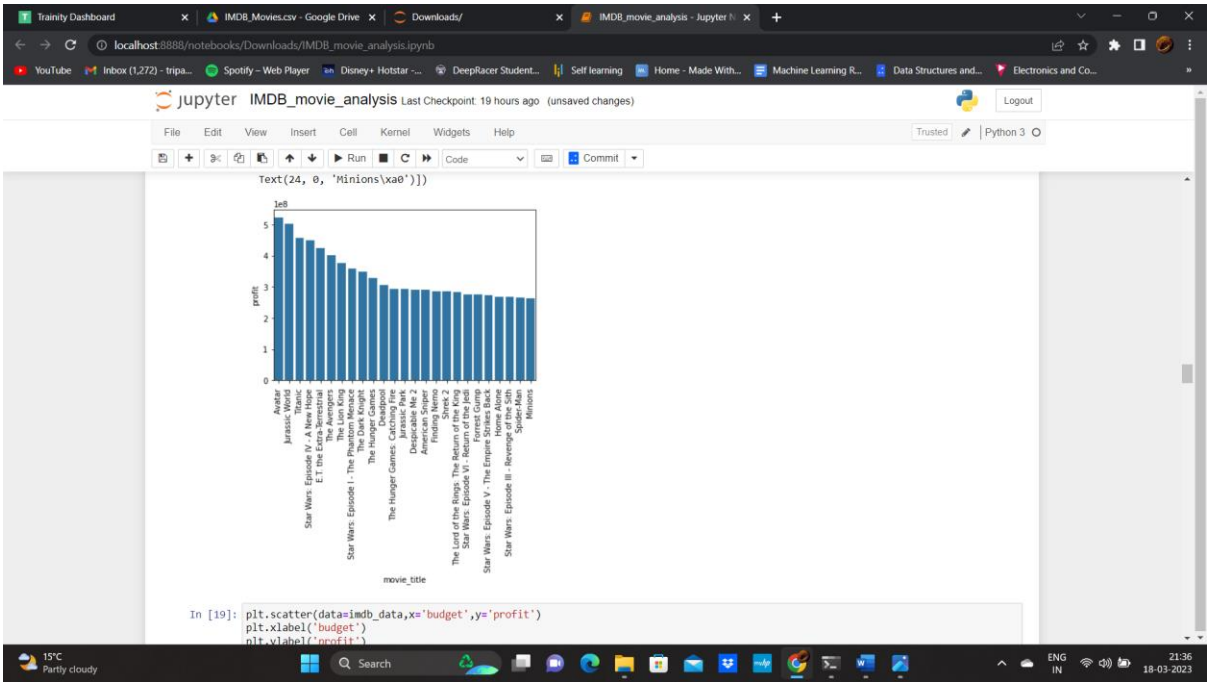
B. **Movies with highest profit:** Create a new column called profit which contains the difference of the two columns: gross and budget. Sort the column using the profit column as reference. Plot profit (y-axis) vs budget (x- axis) and observe the outliers using the appropriate chart type.
Your task: Find the movies with the highest profit?

The following are the top 20 movies with highest profit:

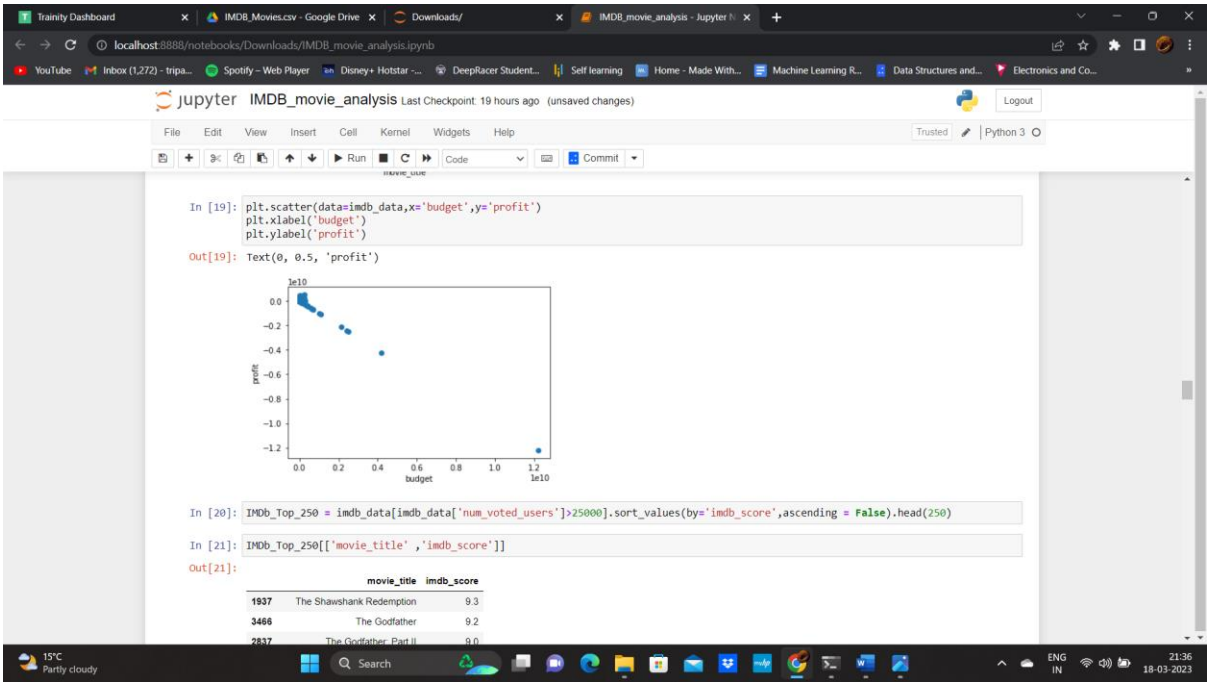


	movie_title	profit	gross	budget
0	Avatar	523505847.0	760505847.0	237000000.0
29	Jurassic World	502177271.0	652177271.0	150000000.0
26	Titanic	458672302.0	658672302.0	200000000.0
3024	Star Wars: Episode IV - A New Hope	449935665.0	460935665.0	11000000.0
3080	E.T. the Extra-Terrestrial	424449459.0	434949459.0	10500000.0
17	The Avengers	403279547.0	623279547.0	220000000.0
509	The Lion King	377783777.0	422783777.0	45000000.0
240	Star Wars: Episode I - The Phantom Menace	359544677.0	474544677.0	115000000.0
66	The Dark Knight	348316061.0	533316061.0	185000000.0
439	The Hunger Games	329999255.0	407999255.0	78000000.0
812	Deadpool	305024263.0	363024263.0	58000000.0
186	The Hunger Games: Catching Fire	294645577.0	424645577.0	130000000.0
697	Jurassic Park	293784000.0	356784000.0	63000000.0
521	Despicable Me 2	292049635.0	368049635.0	76000000.0
814	American Sniper	291323553.0	350123553.0	58800000.0
338	Finding Nemo	286838870.0	380838870.0	94000000.0
582	Shrek 2	286471036.0	436471036.0	150000000.0
339	The Lord of the Rings: The Return of the King	283019252.0	377019252.0	94000000.0
1536	Star Wars: Episode VI - Return of the Jedi	276625409.0	309125409.0	32500000.0
836	Forrest Gump	274691196.0	329691196.0	55000000.0

Bar graph representation of top 20 movies:



The outliers can be observed using scatter plot:

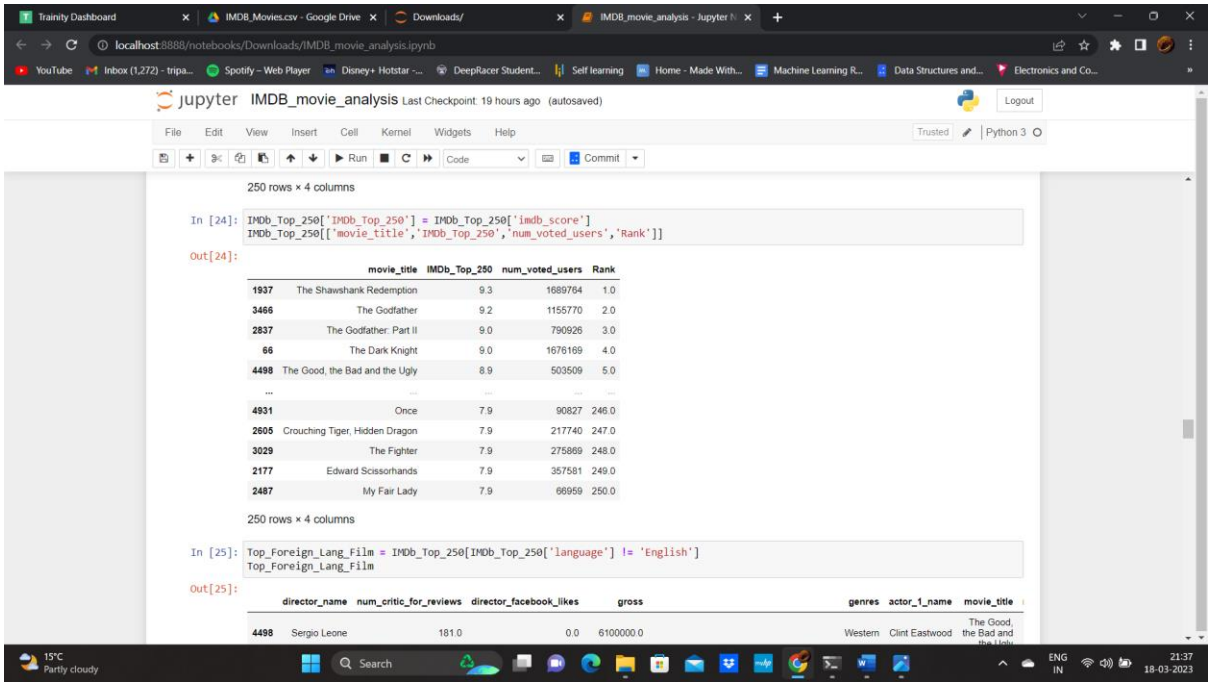


C. **Top 250:** Create a new column IMDb_Top_250 and store the top 250 movies with the highest IMDb Rating (corresponding to the column: imdb_score). Also make sure that for all of these movies, the num_voted_users is greater than 25,000. Also add a Rank column containing the values 1 to 250 indicating the ranks of the

corresponding films.

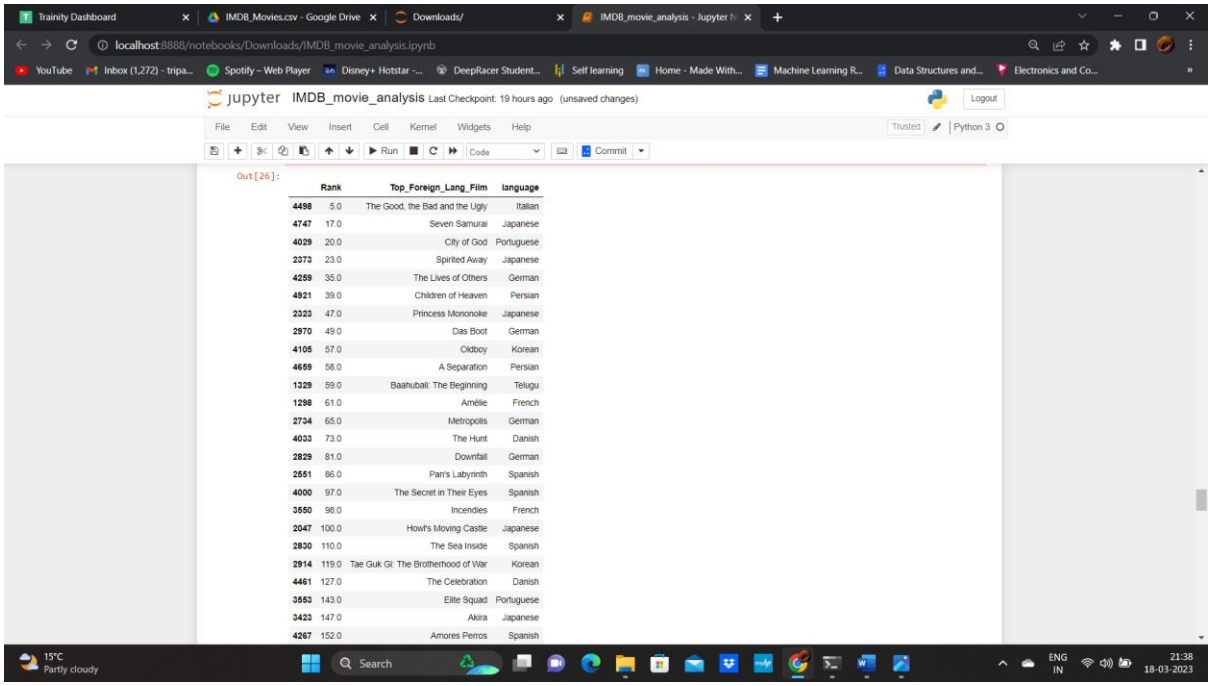
Extract all the movies in the IMDb_Top_250 column which are not in the English language and store them in a new column named Top_Foreign_Lang_Film. You can use your own imagination also!

Your task: Find IMDB Top 250



The following picture shows movie names and IMDb_Top_250 column with top imdb scores along with rank column from 1-250

Top_foreign_language_Films:

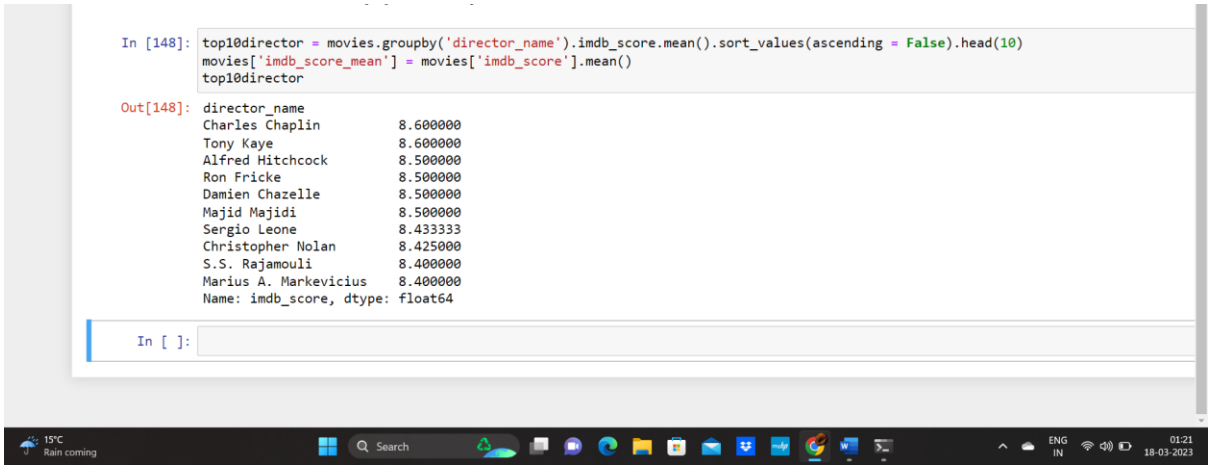


Out[26]:

	Rank	Top_Foreign_Lang_Film	language
4498	5.0	The Good, the Bad and the Ugly	Italian
4747	17.0	Seven Samurai	Japanese
4029	20.0	City of God	Portuguese
2373	23.0	Spirited Away	Japanese
4259	35.0	The Lives of Others	German
4921	39.0	Children of Heaven	Persian
2323	47.0	Princess Mononoke	Japanese
2970	49.0	Das Boot	German
4105	57.0	Oldboy	Korean
4659	58.0	A Separation	Persian
1329	59.0	Baahubali: The Beginning	Telugu
1298	61.0	Amélie	French
2734	65.0	Metropolis	German
4033	73.0	The Hunt	Danish
2829	81.0	Downfall	German
2551	86.0	Pan's Labyrinth	Spanish
4000	97.0	The Secret in Their Eyes	Spanish
3550	98.0	Incendies	French
2047	100.0	Howl's Moving Castle	Japanese
2830	110.0	The Sea Inside	Spanish
2914	119.0	Tae Guk Gi: The Brotherhood of War	Korean
4461	127.0	The Celebration	Danish
3553	143.0	Elite Squad	Portuguese
3423	147.0	Akira	Japanese
4267	152.0	Amores Perros	Spanish

D. Best Directors: TGroup the column using the director_name column.

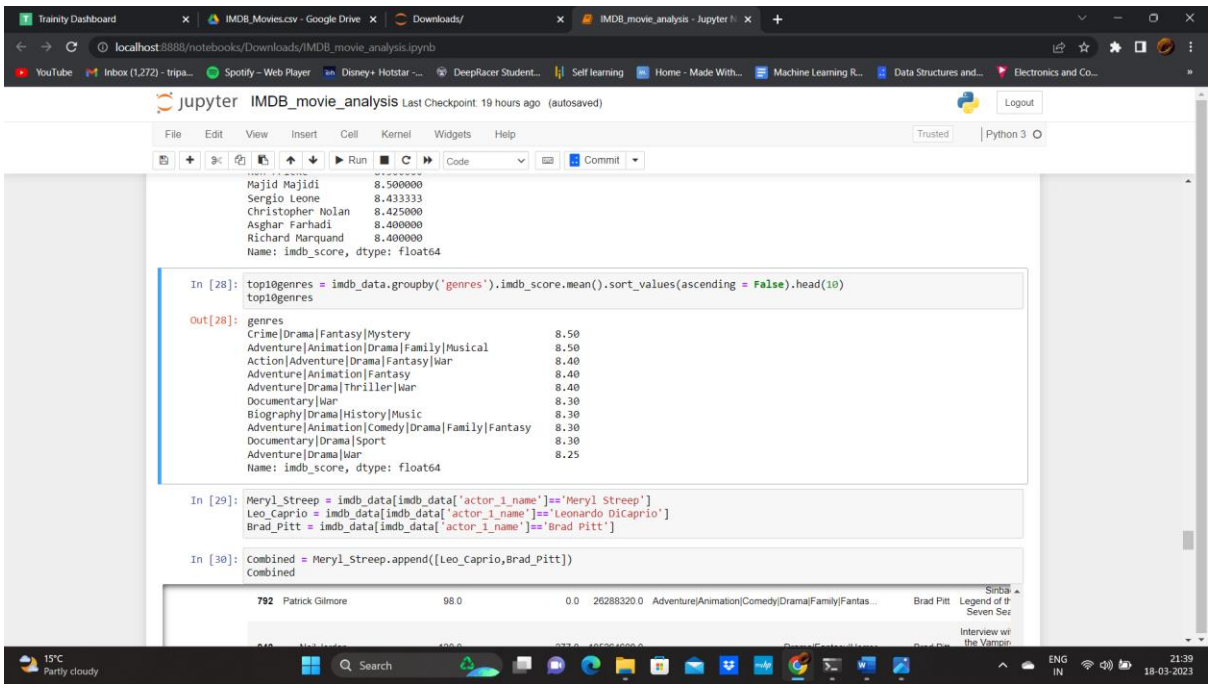
Find out the top 10 directors for whom the mean of imdb_score is the highest and store them in a new column top10director. In case of a tie in IMDb score between two directors, sort them alphabetically.
Your task: Find the best directors



```
In [148]: top10director = movies.groupby('director_name').imdb_score.mean().sort_values(ascending = False).head(10)
          movies['imdb_score_mean'] = movies['imdb_score'].mean()
          top10director

Out[148]: director_name
          Charles Chaplin         8.600000
          Tony Kaye               8.600000
          Alfred Hitchcock        8.500000
          Ron Fricke              8.500000
          Damien Chazelle         8.500000
          Majid Majidi            8.500000
          Sergio Leone            8.433333
          Christopher Nolan       8.425000
          S.S. Rajamouli          8.400000
          Marius A. Markevicius   8.400000
          Name: imdb_score, dtype: float64

In [ ]:
```

The above screenshot shows list of top 10 directors based on the mean of imdb scores.

E. Popular Genres: Perform this step using the knowledge gained while performing previous steps.
Your task: Find popular genres



Following are the top 10a genres .

F. Charts: Create three new columns namely, Meryl_Streep, Leo_Caprio, and Brad_Pitt which contain the movies in which the actors: 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' are the lead actors. Use only the actor_1_name column for extraction. Also, make sure that you use the names 'Meryl Streep', 'Leonardo DiCaprio', and 'Brad Pitt' for the said extraction.

Append the rows of all these columns and store them in a new column named Combined.

Group the combined column using the actor_1_name column.

Find the mean of the num_critic_for_reviews and num_users_for_review and identify the actors which have the highest mean.

Observe the change in number of voted users over decades using a bar chart. Create a column called decade which represents the decade to which every movie belongs to. For example, the title_year year 1923, 1925 should be stored as 1920s. **Sort the** column based on the column decade, group it by decade and find the sum of users voted in each decade. Store this in a new data frame called df_by_decade.

Your task: Find the critic-favorite and audience-favorite actors

1490	Terrence Malick	584.0	0.0	13303319.0	Drama|Fantasy	E
1722	Andrew Dominik	273.0	181.0	3904982.0	Biography|Crime|Drama|History|Western	E
2204	Alejandro G. Iñárritu	285.0	0.0	34300771.0	Drama	E
2333	Angelina Jolie Pitt	131.0	11000.0	531009.0	Drama|Romance	E

```
In [31]: Combined.groupby('actor_1_name').num_critic_for_reviews.mean()

Out[31]: actor_1_name
         Brad Pitt            245.000000
         Leonardo DiCaprio    330.190476
         Meryl Streep         181.454545
         Name: num_critic_for_reviews, dtype: float64

In [32]: Combined.num_user_for_reviews=Combined.num_user_for_reviews.astype('int')

In [33]: Combined.groupby('actor_1_name').num_user_for_reviews.mean()

Out[33]: actor_1_name
```

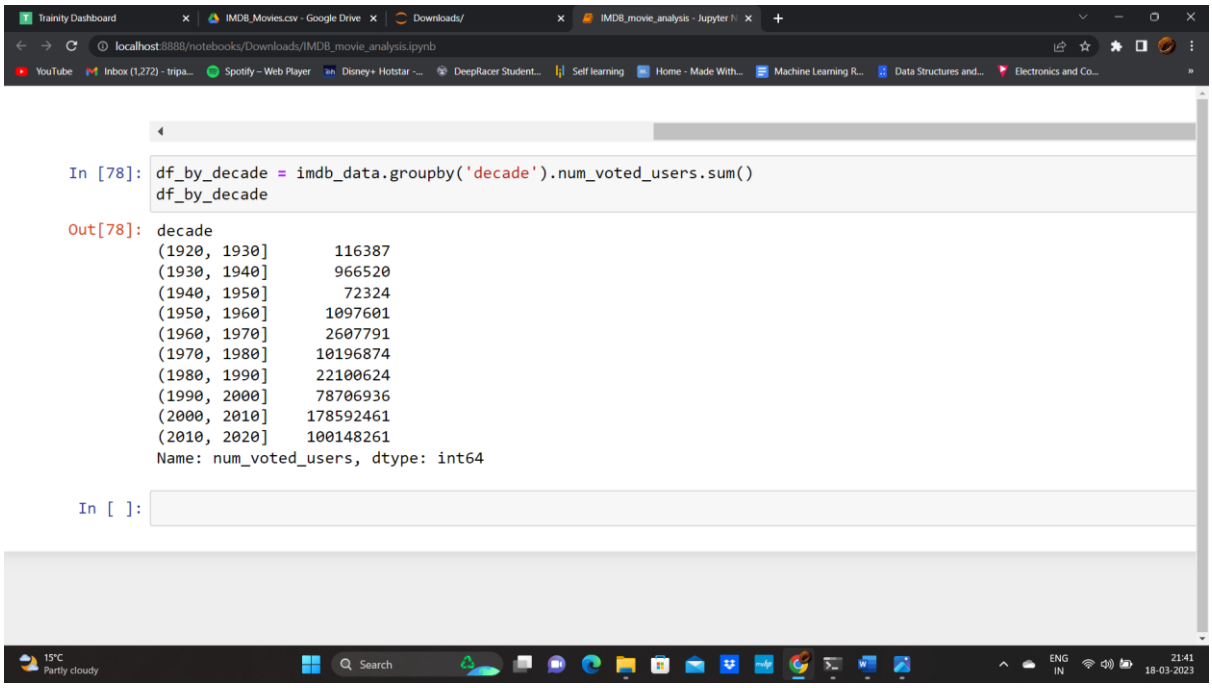
```
In [31]: Combined.groupby('actor_1_name').num_critic_for_reviews.mean()

Out[31]: actor_1_name
         Brad Pitt            245.000000
         Leonardo DiCaprio    330.190476
         Meryl Streep         181.454545
         Name: num_critic_for_reviews, dtype: float64

In [32]: Combined.num_user_for_reviews=Combined.num_user_for_reviews.astype('int')

In [33]: Combined.groupby('actor_1_name').num_user_for_reviews.mean()

Out[33]: actor_1_name
         Brad Pitt            742.352941
         Leonardo DiCaprio    914.476190
         Meryl Streep         297.181818
         Name: num_user_for_reviews, dtype: float64

In [34]: min_year =imdb_data.title_year.min()

In [35]: max_year=imdb_data['title_year'].max()
         int(max_year)+(10 -int(max_year)%10)

Out[35]: 2020
```

Shows the

1) Critic favourite: Leonardo DiCaprio
2) Audience favourite : Leonardo DiCaprio

Observe the change in number of voted users over decades using a bar chart. Create a column called decade which represents the decade to which every movie belongs to. For example, the title_year year 1923, 1925 should be stored as 1920s. Sort the column based on the column decade, group it by decade and find the sum of users voted in each decade. Store this in a new data frame called df_by_decade.

The following ss will show the sum of user votes as per decade:



Result:
Doing this project involved many challenges which made me understand everything and work on it. Using python and its popular libraries for data cleaning and analysis is what in which now I am confident.